goanna *studio*

USER GUIDE – VERSION 2.8.0
Visual Studio EDITION

June 12, 2012

© 2008-2012 Red Lizard Software

Copyright © 2008-2012 Red Lizard Software
All rights reserved.

This document, as well as the software described in it, is provided under license and may only be used or copied in accordance with the terms of such license. The information contained herein is the property of NICTA and is made available under license to Red Lizard Software. It is confidential information as between the Recipient and Red Lizard Software and remains the exclusive property of NICTA. No part of this documentation may be copied, translated, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise without the prior written permission of NICTA.
NICTA does not warrant that this document is error-free.

Red Lizard Software
Australian Technology Park
Level 5, 13 Garden Street
Eveleigh NSW 2015
Australia

Web: http://www.redlizards.com
Support: support@redlizards.com

Contents

1 What's New in Goanna Studio 2.8.0? 2

2 Introduction 2

3 Installation 2
3.1 License Agreement . . . 3
3.2 Installing the Plugin into Visual Studio . . . 3

4 Purchase and License Activation 4
4.1 Machine Challenge Key Generated Licenses . . . 4
4.2 Floating Licenses . . . 6

5 Using Goanna Studio 7
5.1 Sample Code . . . 7
5.2 Running Goanna Studio on a Project . . . 7
5.2.1 Warning Messages . . . 8
5.2.2 Parse Errors . . . 9
5.2.3 Check Descriptions . . . 9
5.2.4 Macro Visualisation . . . 9
5.3 Traces . . . 10
5.4 Goanna Settings Menu . . . 10
5.4.1 Available Goanna Checks . . . 12
5.5 Interprocedural Analysis . . . 12
5.6 Using the Goanna Preprocessor Definition . . . 13
5.7 Using the `assert()` Macro . . . 13
5.8 A Word on False Positives . . . 13

6 Controlling Warnings 14
6.1 Suppressing Warnings . . . 14
6.2 Showing Suppressed Warnings . . . 15
6.3 Unsuppressing Warnings . . . 15
6.4 Exporting Warnings . . . 15

7 Troubleshooting 17
7.1 Known Issues . . . 17
7.2 Support . . . 18

1 What's New in Goanna Studio 2.8.0?

Goanna now has a built-in *false positive elimination* module, that is able to detect more situations when conditional statements and loops *must* be executed based on the values that Goanna can deduce through static analysis. This means that many spurious warnings about uninitialized variables and values not used on some paths will no longer be reported, saving you from having to suppress them manually or change your code to silence them. You don't need to do anything to take advantage of this new feature, it is enabled automatically. The situations where Goanna is able to detect and eliminate false positives are described in more detail under Section 5.8. These situations are being extended all the time, so you can expect even greater precision from Goanna's false positive elimination in the future.

In addition, Goanna now includes additional checks to guard against more potential division-by-zero and double-free errors. These checks are also enabled by default, so, again, you don't have to do anything to get greater coverage of these potential defects. For more information, refer to ATH-div-0-unchk-local, ATH-div-0-unchk-global, and MEM-double-free-some in the Reference Guide.

Further, Goanna Studio now handles user specific configurations better by loading more user configuration files and settings. In particular this loads include paths occasional specific in unusual locations. Also Goanna Studio supports an *ignore list* where users can specify files, paths, and regular expressions that Goanna should not analyse. This allows users to prevent Goanna from analyzing generated files, third party libraries, and other other spacial cases.

2 Introduction

Only a few simple steps are required to get Goanna Studio running on your local machine. This document will guide you through the process of installing and using Goanna Studio. This distribution of Goanna Studio runs on both 32-bit and 64-bit versions of:

- Windows XP
- Windows Vista
- Windows 7

and supports all editions of Visual Studio 2005, 2008, and 2010 (except the Express editions).

3 Installation

Quick Solution Double-click the installer `.msi` file and the installation wizard will guide you through. □

3.1 License Agreement

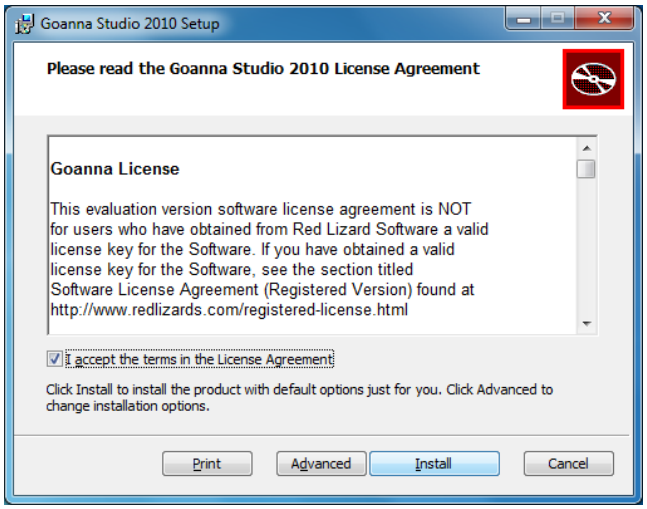
Before installing Goanna Studio, ensure you read the Goanna license agreement, which can be found at
http://redlizards.com/license-term/evaluation-license-agreement/
if you are installing a trial version and
http://redlizards.com/license-term/
if you are installing a registered version of the tool. See Section 4 for how to purchase and install a license.

3.2 Installing the Plugin into Visual Studio

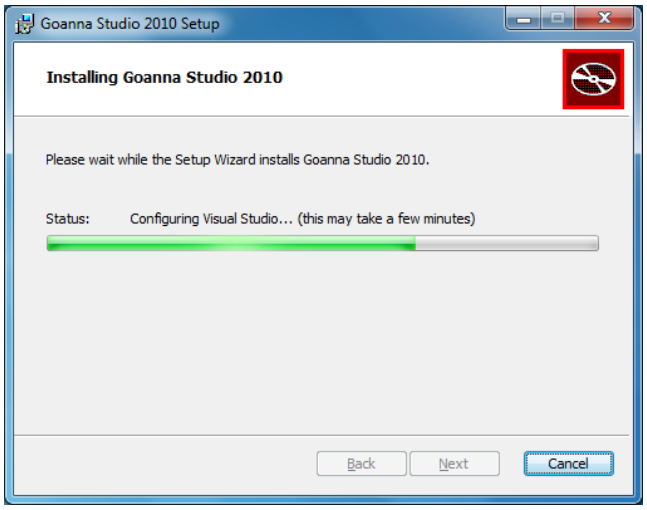
For the installation of Goanna Studio follow the next few steps:

1. Download the Goanna Studio distribution. The package is typically named `GoannaVS-X.msi`, where `X` is the release number.

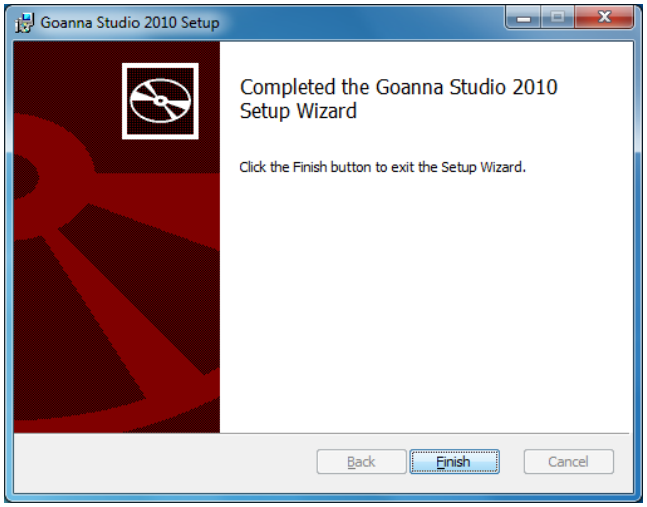
2. Double Click the downloaded `.msi` file, and read and confirm the license agreement.


Goanna Studio 2010 Setup
Please read the Goanna Studio 2010 License Agreement
Goanna License
This evaluation version software license agreement is NOT for users who have obtained from Red Lizard Software a valid license key for the Software. If you have obtained a valid license key for the Software, see the section titled Software License Agreement (Registered Version) found at http://www.redlizards.com/registered-license.html
I accept the terms in the License Agreement
Click Install to install the product with default options just for you. Click Advanced to change installation options.
Print
Advanced
Install
Cancel

3. Follow the installation wizard. Goanna Studio will invoke Visual Studio to complete the integration. This may take **several minutes**.


Goanna Studio 2010 Setup
Installing Goanna Studio 2010
Please wait while the Setup Wizard installs Goanna Studio 2010.
Status: Configuring Visual Studio... (this may take a few minutes)
Back
Next
Cancel

4. At the end of the installation process click `Finish`.



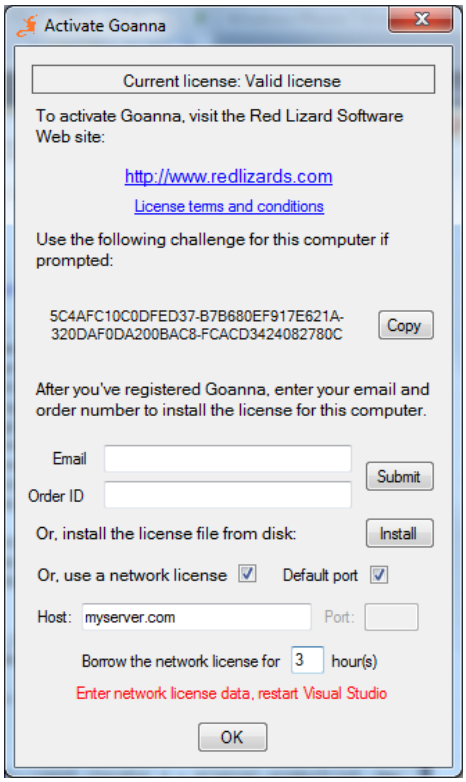
5. Launch Visual Studio.

4 Purchase and License Activation

4.1 Machine Challenge Key Generated Licenses

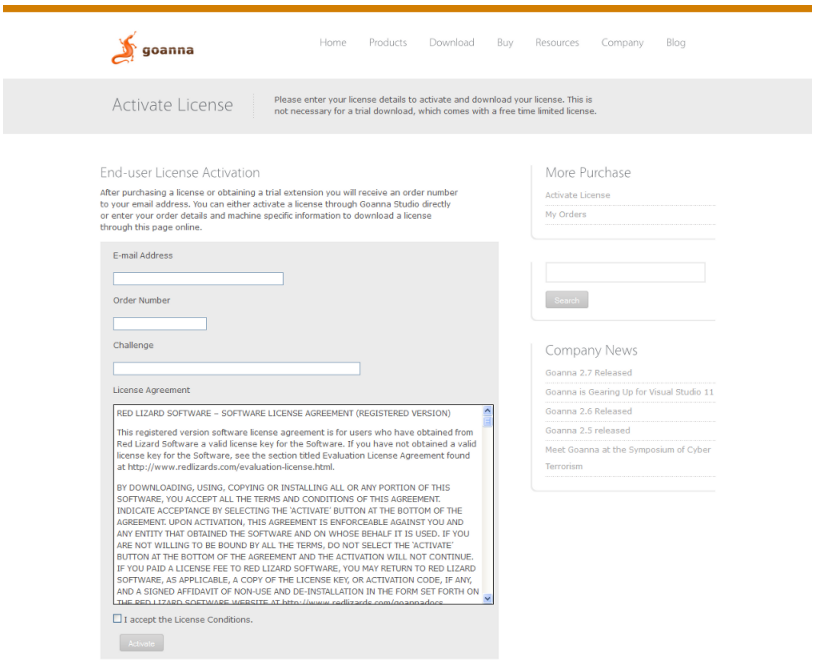
Quick Solution In the `Goanna` menu select `Activate Goanna`. Then choose a method of activating your license from the resulting windows. □

Go to http://redlizards.com/purchase/ and follow the purchase instructions. Once you purchase one or multiple licenses you will receive an email with the details of your purchase, including the order number. This gives you the right to activate a matching number of seats. For each seat go to the Visual Studio `Goanna` menu and select `Activate Goanna`, which will show you the `Activate Goanna` dialog.



Then do one of the following:

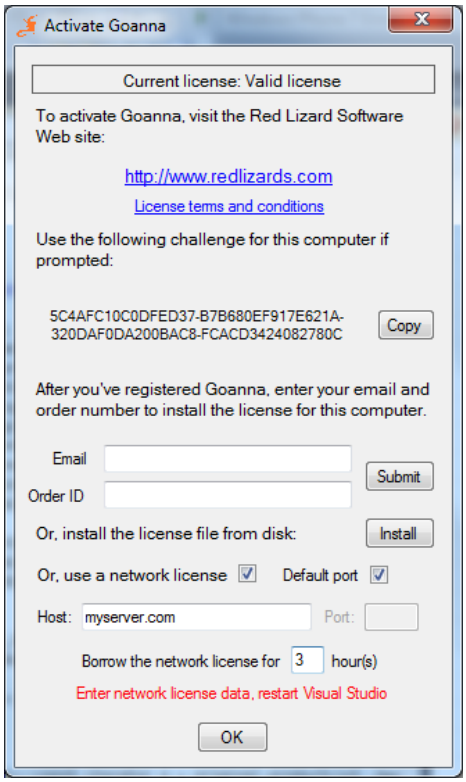
- enter your email address and the order number from the time of purchase, or
- if you are using a network license, check the `Use a Network License` option, and type the address of your network license server in the `Host` box, or
- use your web browser to go to the Goanna *activation page* `http://www.redlizards.com/purchase/activate-license/` . Enter the details requested, the challenge field requires the challenge key found on the `Activate Goanna` dialog. Then, manually upload the resulting license file using the `Install` button on the `Activate Goanna` dialog.


goanna
Home
Products
Download
Buy
Resources
Company
Blog
Activate License
Please enter your license details to activate and download your license. This is not necessary for a trial download, which comes with a free time limited license.
End-user License Activation
After purchasing a license or obtaining a trial extension you will receive an order number to your email address. You can either activate a license through Goanna Studio directly or enter your order details and machine specific information to download a license through this page online.
E-mail Address
Order Number
Challenge
License Agreement
RED LIZARD SOFTWARE – SOFTWARE LICENSE AGREEMENT (REGISTERED VERSION)
This registered version software license agreement is for users who have obtained from Red Lizard Software a valid license key for the Software. If you have not obtained a valid license key for the Software, see the section titled Evaluation License Agreement found at http://www.redlizards.com/evaluation-license.html.
BY DOWNLOADING, USING, COPYING OR INSTALLING ALL OR ANY PORTION OF THIS SOFTWARE, YOU ACCEPT ALL THE TERMS AND CONDITIONS OF THIS AGREEMENT. INDICATE ACCEPTANCE BY SELECTING THE 'ACTIVATE' BUTTON AT THE BOTTOM OF THE AGREEMENT. UPON ACTIVATION, THIS AGREEMENT IS ENFORCEABLE AGAINST YOU AND ANY ENTITY THAT OBTAINED THE SOFTWARE AND ON WHOSE BEHALF IT IS USED. IF YOU ARE NOT WILLING TO BE BOUND BY ALL THE TERMS, DO NOT SELECT THE 'ACTIVATE' BUTTON AT THE BOTTOM OF THE AGREEMENT AND THE ACTIVATION WILL NOT CONTINUE. IF YOU PAID A LICENSE FEE TO RED LIZARD SOFTWARE, YOU MAY RETURN TO RED LIZARD SOFTWARE, AS APPLICABLE, A COPY OF THE LICENSE KEY, OR ACTIVATION CODE, IF ANY, AND A SIGNED AFFIDAVIT OF NON-USE AND DE-INSTALLATION IN THE FORM SET FORTH ON
I accept the License Conditions.
Activate
More Purchase
Activate License
My Orders
Search
Company News
Goanna 2.7 Released
Goanna is Gearing Up for Visual Studio 11
Goanna 2.6 Released
Goanna 2.5 released
Meet Goanna at the Symposium of Cyber Terrorism

In order to activate the license after installing or borrowing it, you will need to restart Visual Studio.

4.2 Floating Licenses

If you are using a License network server, you can borrow a license by going to `Goanna->Activate Goanna`. On the `Activate Goanna` dialog windows, you can enter the address of your license server, and the borrow duration.


Activate Goanna
Current license: Valid license
To activate Goanna, visit the Red Lizard Software Web site:
http://www.redlizards.com
License terms and conditions
Use the following challenge for this computer if prompted:
5C4AFC10C0DFED37-B7B680EF917E621A-320DAF0DA200BAC8-FCACD3424082780C
Copy
After you've registered Goanna, enter your email and order number to install the license for this computer.
Email
Order ID
Submit
Or, install the license file from disk:
Install
Or, use a network license
Default port
Host: myserver.com
Port:
Borrow the network license for 3 hour(s)
Enter network license data, restart Visual Studio
OK

5 Using Goanna Studio

5.1 Sample Code

A package containing a number of sample **C**/**C++** files is available on our website. Go to `http://www.redlizards.com/resources/example-code/` and download the `Visual Studio Sample Code` package corresponding to your version of VISUAL STUDIO. The files in this package may be useful for practicing using Goanna Studio, or ensuring that Goanna is working correctly.

A project file and solution file for VISUAL STUDIO are included in this directory. Open either of these files to open the solution.

Once the project has loaded, you can try analyzing it with Goanna Studio.

5.2 Running Goanna Studio on a Project

Quick Solution Select the files, project or solution you want analyzed in the Project Explorer and click the Goanna icon in the toolbar. Goanna Studio will automatically display warnings, if there are any. □

Goanna Studio can be invoked on a project or a solution by selecting the appropriate entry from the `Goanna` menu, where the Goanna icon appears (see Fig. 1). You can also select several files and right-click on these. The `Goanna` option in the context menu allows you to run the tool over the selected files

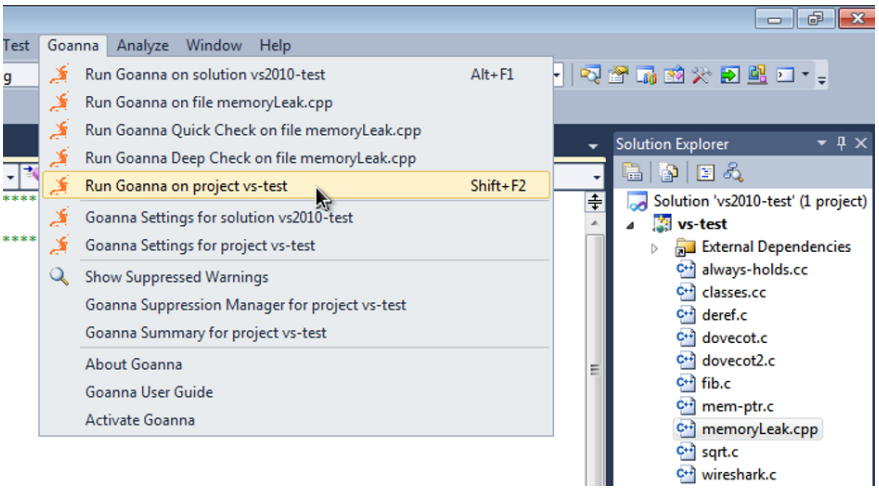


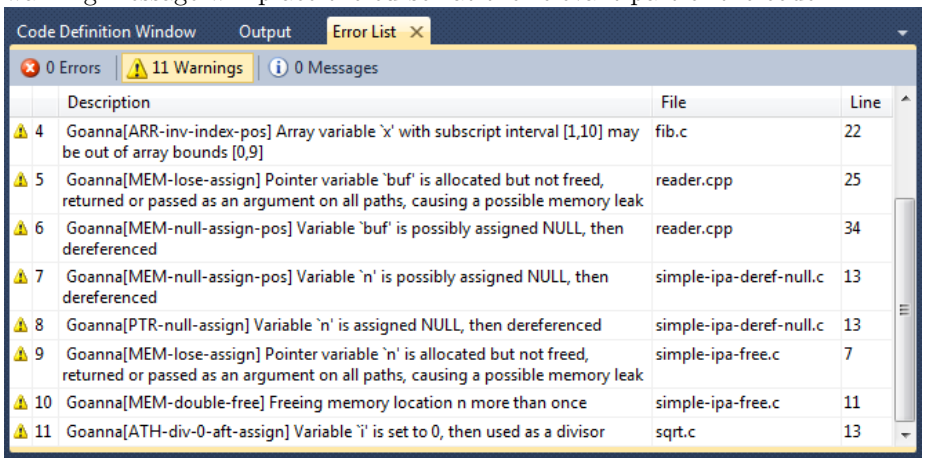
Figure 1: Analysis of an entire project.

only (Fig. 2). Goanna can be called in the same way from the context menu of a project or a solution in the Project Explorer.

Goanna Studio has pre-defined *Quick Check* and *Deep Check* configurations. A Quick Check turns on all the Goanna checks that do not require significant computation. A Deep Check runs every Goanna check and includes deep analysis including interproecural analysis (see Section 5.5 for details).

5.2.1 Warning Messages

Any warnings from Goanna Studio will appear in the warnings list. This list will also identify the file and line number associated with each warning, in much the same way as the compiler's warning and error messages. Double-clicking a warning message will place the cursor at the relevant part of the code.



	Description	File	Line
4	Goanna[ARR-inv-index-pos] Array variable `x' with subscript interval [1,10] may be out of array bounds [0,9]	fib.c	22
5	Goanna[MEM-lose-assign] Pointer variable `buf' is allocated but not freed, returned or passed as an argument on all paths, causing a possible memory leak	reader.cpp	25
6	Goanna[MEM-null-assign-pos] Variable `buf' is possibly assigned NULL, then dereferenced	reader.cpp	34
7	Goanna[MEM-null-assign-pos] Variable `n' is possibly assigned NULL, then dereferenced	simple-ipa-deref-null.c	13
8	Goanna[PTR-null-assign] Variable `n' is assigned NULL, then dereferenced	simple-ipa-deref-null.c	13
9	Goanna[MEM-lose-assign] Pointer variable `n' is allocated but not freed, returned or passed as an argument on all paths, causing a possible memory leak	simple-ipa-free.c	7
10	Goanna[MEM-double-free] Freeing memory location n more than once	simple-ipa-free.c	11
11	Goanna[ATH-div-0-aft-assign] Variable `i' is set to 0, then used as a divisor	sqrt.c	13

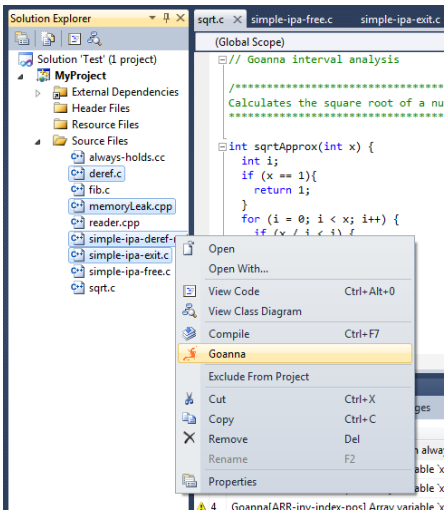


Figure 2: Analysis of selected files.

5.2.2 Parse Errors

Goanna identifies syntactically erroneous code, ensuring that analysis only takes place on syntactically correct code. If a file contains syntax errors Goanna will simply skip analysis of that file. (This will be in the `Output` panel from Goanna but not raised in the `Error List` by default.) However, in this case there are two ways to see the parsing errors.

When parsing errors occur a new menu item appears from the `Goanna` menu; `Show Parse Errors for ...` Selecting this menu item will then display the Goanna parsing errors in the `Error List`.

The alternative is to go to the settings for your solution or project and turn on *Verbose output* (see section 5.4).

5.2.3 Check Descriptions

The Goanna Checks tab allow you to view an online version of the check descriptions that appear in the companion reference manual *Goanna Reference Manual*. Simply right-click a warning message and select `Describe Check` from the context menu to view a detailed description of the check, including simplified code examples.

5.2.4 Macro Visualisation

Sometimes it is difficult to determine the reason for a particular warning, because the code invokes a **C** preprocessor (**C++**) macro. Visual Studio offers a tooltip window showing the macro definition, but that may not be enough for effective debugging.

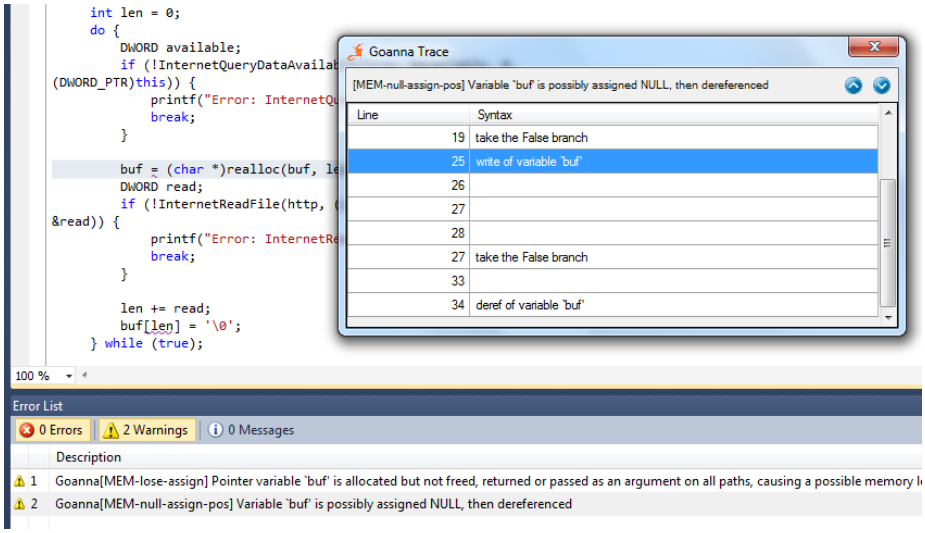
The Visual Studio edition of Goanna Studio can show the expansion of **C++** macros. Right-click anywhere in a Visual Studio code window to get a context menu, and choose `Show macros`. Each **C++** macro in the source file

will be highlighted with a blue underline. Hover over the macro to show its expansion in a tooltip window.

While macros are being displayed, any purple underlines indicating Goanna warnings are hidden. From the same context menu you used to show the macros, choose `Clear macros` to remove the blue macro underlines and restore the purple warning underlines.

5.3 Traces

The Goanna trace window will allow you to trace through the path of execution leading to a warning message. This may help with the debugging of larger functions. Simply right-click the warning message and select `Show trace` from the context menu to view the Goanna Trace window.



5.4 Goanna Settings Menu

Under the Goanna menu there are settings for `solution` or `project`. These have the same option but apply to the solution or project as labelled. A description of these Goanna Studio options is as follows:

- *Ignored paths*: This is a list of files, paths and regular expressions that Goanna should not include in analysis. The drop down menu to the right allows the addition of new files, paths, or regular expressions. The `Remove Selected` button removes all selected items from the ignored paths list.

- *Additional arguments*: this allows you to pass arguments to the Goanna executable. Most users will not need to use this field, because Goanna Studio synthesizes the correct arguments from project files. In some cases, though, a user may wish to pass arguments not found in a project file. When running Goanna, you may want to search for `#include` files in particular directories using the -I flag: `-I C:\SomeIncludePath`. A full list of the possible arguments to the Goanna executable is beyond the scope

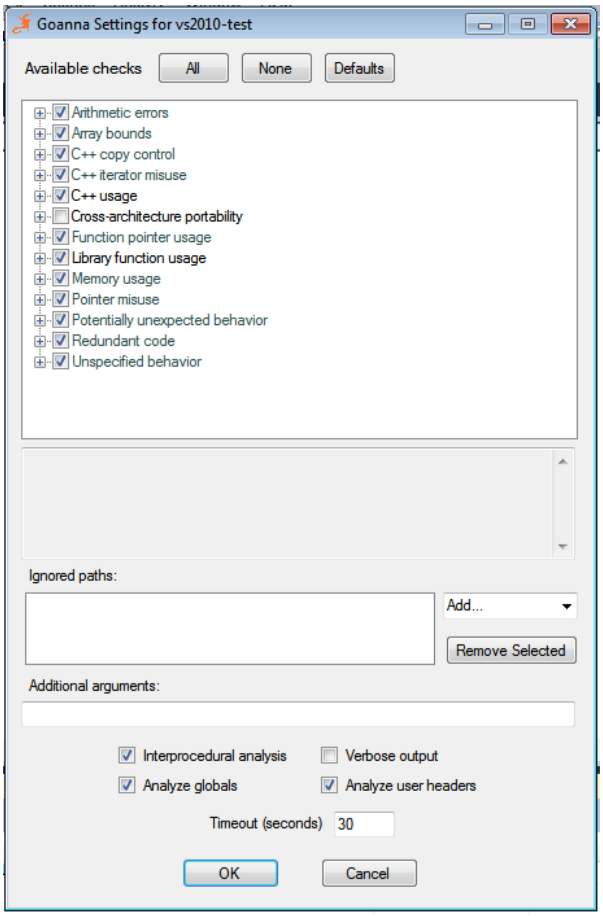


Figure 3: Goanna Settings Menu.

of this document. Please contact us at `mailto:support@redlizards.com` for more information on this topic, if needed.

- *Interprocedural analysis*: This enables interprocedural analysis. See section 5.5 later in this document for a detailed description of this feature.
- *Analyze globals*: Instructs Goanna to include global variables in its analysis.
- *Verbose output*: This causes Goanna to output more detailed information including some debugging data.
- *Analyze user headers*: If this is unchecked it instructs Goanna to ignore functions that are defined in header files.
- *Timeout*: This is a per-file timeout in seconds. Goanna Studio will run for `<seconds>` seconds and terminate with the issues found until then. A timeout of 0 will cause Goanna to analyze the entire file with no time limit.

5.4.1 Available Goanna Checks

Quick Solution These checkboxes offer the option to select the checks that Goanna performs. □

To configure and select the checks that Goanna Studio performs in each run, select or unselect specific checks. For a description of a specific check simply click on the check code and short description next to the checkbox. Details and examples of the checks can be found in the companion reference manual *Goanna Reference Manual*.

5.5 Interprocedural Analysis

In order to improve the coverage and accuracy of the tool, Goanna can perform some *interprocedural analysis*. That is, it can find some bugs that occur as a result of calls between functions.

Once enabled, interprocedural analysis implies that data passed from one function to another, such as integer values, are also propagated in the analysis. For example, this enables Goanna to detect certain cases where a `NULL` pointer is passed from one function to another, and dereferenced at either one.

An example of what interprocedural analysis can find can be seen in the sample of function `myAlloc`.

```
1  #define NULL (void*)0
2
3  void *myAlloc(int param){
4    void *p = malloc(param);
5    if (p)
6       return p;
7     else
8       return NULL;
9  }
10
11 int main(int argc, char ** argv) {
12   int * n;
13   n = (int*)myAlloc(sizeof(int) * 10);
14   n[0] = 5; // this may be a dereference of NULL
15   return *n;
16 }
```

Here, Goanna learns that `myAlloc` may return `NULL` at line 8. This means that when the return value of `myAlloc` is stored at `n` at line 14, Goanna knows this value may be `NULL`. Therefore, line 16 may be dereferencing a `NULL` pointer, and Goanna will warn accordingly.

Please note the following:

1. Interprocedural analysis is not limited to a specific set of checks, but rather enhances the potential precision of most checks. It comes, however, with additional computational overhead. If you require a quick analysis without much depth you should leave the box unchecked.

2. Sometimes, interprocedural effects need more than one analysis run to be fully explored. This is especially true for recursive functions or calls to functions in other files. This is implemented in this way for speed optimization.

5.6 Using the Goanna Preprocessor Definition

Goanna has a built-in preprocessor definition, defined by the macro

```
1  #define _GOANNA 1
```

This allows code to be explicitly included in or excluded from analysis by Goanna. For example:

```
1  #ifdef _GOANNA
2  //Code only to be included while the program is being
       analysed
3  #endif
4  #ifndef _GOANNA
5  //Code not to be analyzed by Goanna
6  #endif
```

5.7 Using the `assert()` Macro

Goanna can sometimes use information provided by `assert()` to refine its analysis of numerical and pointer values. It does this by using `assert` statements as assumptions for value ranges and pointer validity.

For example, in the code below:

```
1    void my_fun(void) {
2      int my_array[20];
3      int x = rand();
4
5      assert(x == 10);
6      f(my_array[x]);
7    }
```

the `assert()` means that the array reference must be in-bounds, even though the index variable `x` has a randomly-assigned value. Therefore, Goanna does not issue an out-of-bounds warning.

5.8 A Word on False Positives

Goanna considers all possible execution paths in your program, and will warn you if it finds potential defects (such as use of an uninitialised variable) that occur only on particular execution paths and not others. But sometimes, the execution path leading to a potential defect is actually not possible when the program is executed. If Goanna is able to deduce this through static analysis, then it won't warn you. But if it can't, then you may receive a spurious warning for a defect that isn't really there. Such warnings are called *false positives*.

So for example in the following simple code:

```
1  int n;
2  int size = 10;
3  for (int i = 0; i <= size; i++) {
4    n = i;
5  }
6  printf("%d\n", n);
```

Goanna knows that the body of this loop is executed 10 times and that `n` will be well and truly initialised after the loop, so it does not warn you about use of an uninitialised variable in the `printf` statement.

Some false positives can only be eliminated if Goanna is performing inter-procedural analysis. For example, in the following code:

```
1  void test(int max) {
2    int n;
3    int size = max;
4    for (int i = 0; i <= size; i++) {
5      n = i;
6    }
7    printf("%d\n", n);
8  }
9  int main() {
10     test(10);
11 }
```

Goanna will only suppress the warning if inter-procedural analysis is enabled. If it is not, then you will see a false positive warning.

Goanna currently does not evaluate boolean conditions and this may lead to false positives that are not eliminated automatically. Consider the code below:

```
1  void foo(bool b)
2  {
3    MyClass *p;
4
5    if (b)
6      p = new MyClass;
7
8    if (b)
9      *p;
10 }
```

Goanna will issue a warning that `p` may not have been initialized before its dereference. The two `if` conditions are treated separately, and Goanna explores all four paths of control: true/true, true/false, false/true and false/false. As it turns out, one path false/true may lead to uninitialized pointer dereference. This path is however not feasible and thus the warning is a false positive.

Such false positives may be suppressed with `asserts` (sec. 5.7) or with Warning Filtering (see below).

6 Controlling Warnings

For a number of reasons, you may wish to ignore specific warning messageses without modifying your check settings. Goanna Studio allows you to easily hide and show warnings to help keep emphasis on the most critical bugs.

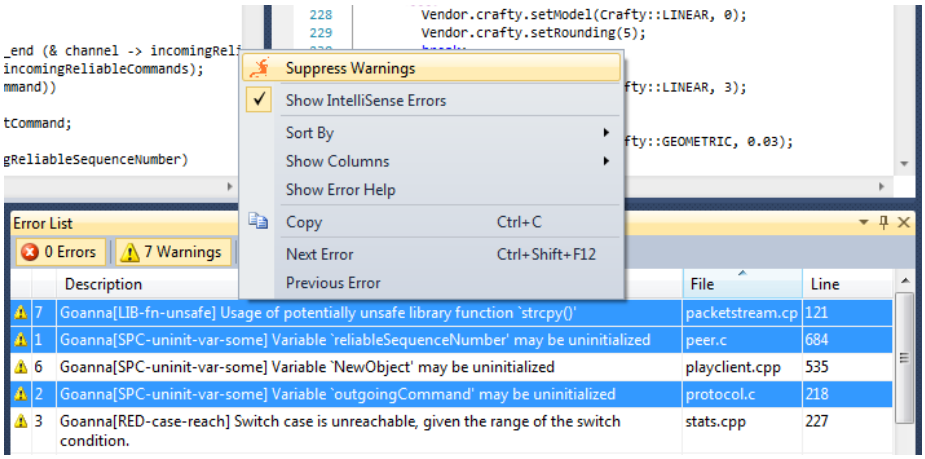
6.1 Suppressing Warnings

Quick Solution Right-click a warning and select `Suppress Warning` from the context menu. To view all suppressed warnings, click the `Show Suppressed Warnings` button in the toolbar. From there, you can unsuppress warnings in the same way. □

To hide a warning, right-click it in the `Goanna Warnings` list, and select `Suppress Warning` from the context menu.

Goanna takes special care to ensure that warning suppression never silently conceals actual regressions (bugs that are fixed but inadvertently reintroduced) or unrelated bugs. If you suppress a warning, and later modify the function or any other code having an effect on it, Goanna will notice that the code has

been edited, and re-issue the warning. Internally Goanna uses a hash of the actual syntax to determine if the code has changed, so only significant changes will cause the warning to reappear, not just modifying the file or changing whitespace or comments.



6.2 Showing Suppressed Warnings

When you ask Goanna Studio to ignore a warning, it will not be removed permanently from the database. The list of Goanna warnings in the `Goanna Warnings` tab can be toggled, to show either the suppressed or unsuppressed warnings. Click the `Show Suppressed Warnings` button on the right-hand side of the `Goanna Warnings` tab to toggle which warnings are shown.

6.3 Unsuppressing Warnings

To unsuppress a warning, ensure you are showing suppressed warnings in the `Warnings` tab. Select the warning you wish to unsuppress, right-click on it, and select `Unsuppress Warning` from the context menu.

You can unsuppress multiple warnings at once. To do this, select the warnings in the same way as you would to suppress them, right-click on one of the selected warnings, and select `Unsuppress Warnings` from the context menu.

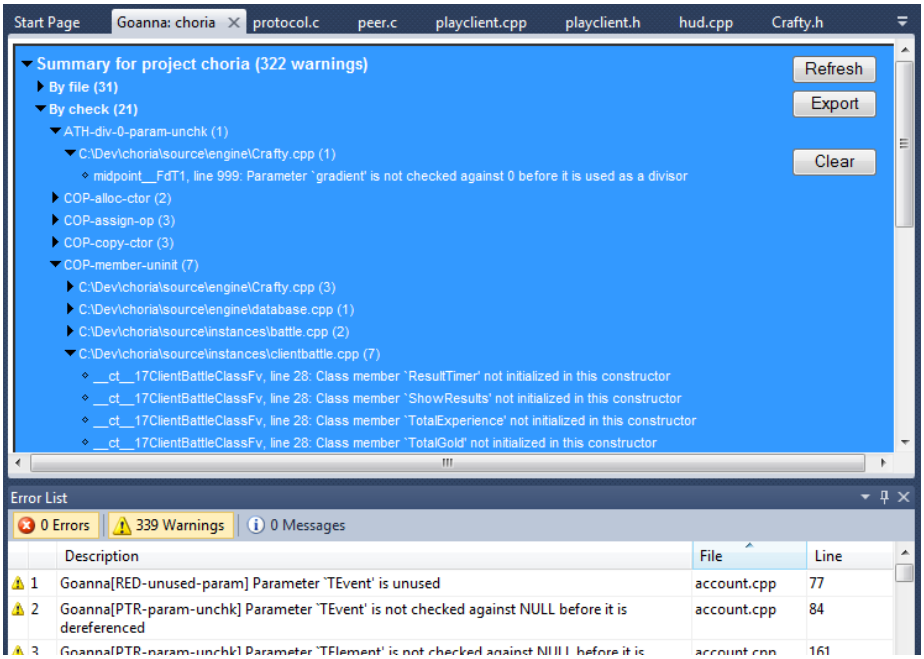
6.4 Exporting Warnings

Quick Solution Select `Goanna Summary` from the `Goanna` menu to open the `Goanna Summary` window. Click the `Export` button to export the warnings in comma-separated-value (CSV) format. If you use Microsoft Excel to load the exported warnings, note that in some countries, Excel expects semicolons instead of commas to separate values. You can save the warnings file and globally replace the text "," with ";" before loading it into Excel. Alternatively, rename the warnings file with a `.txt` extension, and Excel will prompt you which separator to use. □

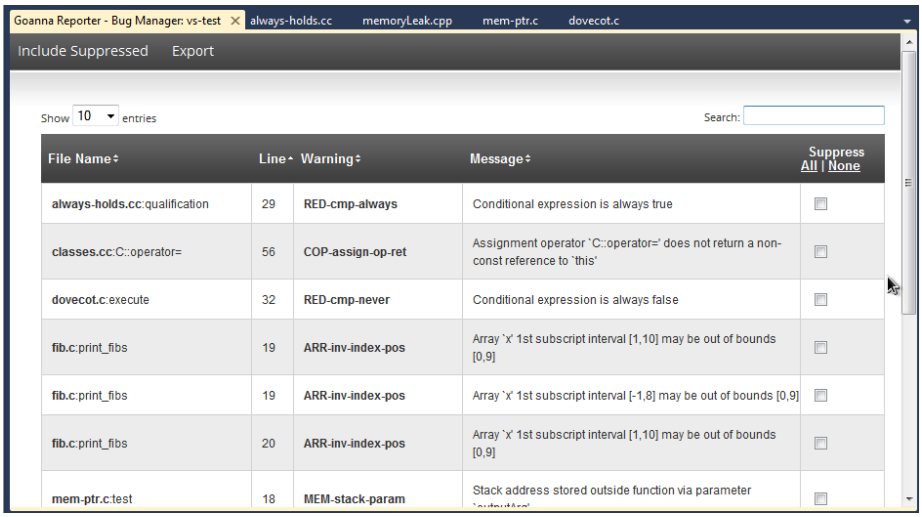
The `Goanna Summary` window will provide you with a method of organizing Goanna's warning messages, and also a method of exporting them. Under `Goanna`, choose `Goanna Summary`.

Two sets of drop-down lists will appear: `By File` and `By Check`. These can be expanded to list the warnings of the project either by file name or check name alphabetically. You can use the `Goanna Summary` window to export the warnings. Clicking on `Export` downloads the warnings of the project in CSV (Comma Separated Value) format.

You can also use this window to clear Goanna's database. This will remove all warnings that have been identified, and reset all stored data. This means the compounded data from interprocedural analysis runs will be deleted. To clear the database, click on `Clear`.



An alternative way to manage warnings is the `Goanna Suppression Manager` that lists, sorts, and toggles suppression of warnings. Here you can sort warnings by file name, line number, warning type, and warning message. You can also filter the results to show just the warnings you are interested in. Each warning, or page of displayed warnings, can be suppressed to prevent it showing up in future Goanna results (details on suppression are discussed starting in Section 6.1).


Goanna Reporter - Bug Manager: vs-test
always-holds.cc
memoryLeak.cpp
mem-ptr.c
dovecot.c
Include Suppressed
Export
Show 10 entries
Search:
File Name
Line
Warning
Message
Suppress
All | None
always-holds.cc:qualification 29 RED-cmp-always Conditional expression is always true
classes.cc:C::operator= 56 COP-assign-op-ret Assignment operator `C::operator=' does not return a non-const reference to `this'
dovecot.c:execute 32 RED-cmp-never Conditional expression is always false
fib.c:print_fibs 19 ARR-inv-index-pos Array `x' 1st subscript interval [1,10] may be out of bounds [0,9]
fib.c:print_fibs 19 ARR-inv-index-pos Array `x' 1st subscript interval [-1,8] may be out of bounds [0,9]
fib.c:print_fibs 20 ARR-inv-index-pos Array `x' 1st subscript interval [1,10] may be out of bounds [0,9]
mem-ptr.c:test 18 MEM-stack-param Stack address stored outside function via parameter

Similar to the `Goanna Summary`, warnings can be explorted in CSV format via the `Export` button.

7 Troubleshooting

This section describes a number of known issues and solutions to them. Should you find more issues you think we should know about, please contact us at mailto:support@redlizards.com.

7.1 Known Issues

Managed Code

Managed code is not currently supported by Goanna Studio.

Generated Files

Goanna Studio cannot distinguish between generated files and the hand-coded files that generate them, and will issue warnings in the files in which they are found, in every instance they are found.

This means that large generated files may yield a large number of warnings stemming from a single bug in the hand-coded file which generated it. This will cause a warning for every manifestation of this bug in the generated code.

C++11 standards support (Goanna Studio 2010 Only)

Goanna Studio 2010 is not able to support all C++11 features provided within Visual Studio 2010. Some features, such as the `auto` keyword are supported, however more advanced features like inline namespaces are not supported and Goanna Studio is not able to parse them. Future releases will support more features.

The #import Directive

When using the #import directive to import a dynamic library (.dll) or type library (.tlb), it is necessary to compile the file containing the directive before it is analysed by Goanna Studio. Compilation generates a header file with the extension .tlh.

Then in the source code where you perform your import, you can use the `_GOANNA` macro to conditionally include the .tlh file. For example:

```
1  #ifdef _GOANNA
2  #include ".\Debug\msxml3.tlh"
3  #else
4  #import "msxml3.dll"
5  #endif
```

Precompiled Header Through Files

Goanna Studio does not currently handle 'precompiled header through' files, used with the MSVC option `/Yu`, if the specified file does not exist. The third-party **C**/**C++** parser we use does not handle precompiled header files in the way given by the `/Yu` option.

7.2 Support

Some unforeseen issues might occur while using Goanna Studio, depending on your particular compiler version, include files, hardware, undocumented features in **C**/**C++** or some combination of these. If you come across any problems, please contact mailto:support@redlizards.com or visit out support page at http://redlizards.com/resources/.

Index

License, 3